2002

An agent-based approach for information source selection under distributed information environments

Hui Yang
University of Wollongong

An Agent-Based Approach for Information Source Selection under Distributed Information Environments

A thesis submitted in partial fulfillment of the

requirements for the award of the degree

Master of Science (Honours)

(Computer Science)

from

THE UNIVERSITY OF WOLLONGONG

by

Hui Yang, BSc, MSc.

School of Information Technology and Computer Science

March 31, 2002

CERTIFICATION

I, Hui Yang, declare that this thesis, submitted in partial fulfillment of the requirements for the award of Master of Science (Honours), in the School of Information Technology and Computer Science, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Hui Yang

31 March, 2002

Abstract

With inexhaustible amount of information available online and heterogeneously distributed over the Internet, distributed information retrieval (DIR) has become an important area in information retrieval (IR) research in recent years, which has raised a new set of issues specific to DIR including information source selection, query processing and result fusion. Information source selection is how to select a subset of the distributed collections, which are most likely to contain relevant documents for the current query rather than broadcasting every request to every collection in the system.

Intelligent agents offer promising solutions to the current explosion on the Internet and the problem of information retrieval. They have potentiality of mitigating the complexity of information retrieval and management by exhibiting some key attributes such as autonomy, intelligence and adaptability. These specific features help intelligent agents to act as an assistant/assistants to the user in carrying out the task of information retrieval.

In this thesis, we introduce a new approach, agent-based intelligent information source selection, which is an alternative way for overcoming the problem of information selection from distributed information sources by using three artificial intelligence techniques, including *query expansion* with *a Naive Bayes text classifier*, intelligent information selection with *case-based reasoning* and adaptation to the dynamic web environment with *reinforcement learning*. My contribution to this research is to propose an intelligent environment where the Analysis Agent, Case-

Matching Agent and Learning Agent, these three major agents iteratively work together to locate the most appropriate information sources to search so as to effectively and efficiently satisfy the user's expectation.

We have finished the implementation of the first component – Analysis Agent. The experimental results show that it is possible to improve the effectiveness on both selection and retrieval stages in a distributed searching environment by using query expansion with *a Naive Bayes text classifier.*

Acknowledgements

First and foremost, I would like to express my sincere thanks to my supervisor, Dr. Minjie Zhang for her contribution in the development of my research and her guidance during the course of master study. Without her broad vision, deep insight, and strong encouragement, this work would not have been possible.

I would like to thank Dr. Xiaohua Yang for his valuable comments and suggestions on distributed information retrieval.

I would like to thank all my friends I have met in Australia. Without them, the last one-year would have been unbearable.

In addition, my thanks go to my examiners, Professor Aditya Ghose and Dr. Xiangjian He. Their suggestions and comments are quite explicit and helpful for improving the quality of my thesis. My thanks go to Mr. Jesson Williams for his help in proofreading my thesis to improve the written presentation of the final copy.

Finally, I would also like to express my deep gratitude to my family who give me their love, understanding and absolute faith in my abilities. I wish they are proud of me!

Publications

The followings are a list of my research papers that have been already published during my Master (Honours) study that is to ended by the completion of this thesis.

- Hui Yang, Minjie Zhang and Xiaohua Yang, IISS: A Framework for Intelligent Information Source Selection on the Web. In *Proceedings of International Conference on Artificial Neural Networks and Expert Systems*, Dunedin, 2001, pp. 209-215.
- Hui Yang and Minjie Zhang, Legal Aspects of the Application of Agent-Based Information Retrieval on the Internet. In *Proceedings of International Conference on Information Technology and the Emerging Law,* Wollongong, 2001, pp. 42-48.
- Xiaohua Yang, Hui Yang and Minjie Zhang, Fusion Methods Based on Common Order Invaribility for Meta Search Engine Systems. In *Proceedings of International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, Las Vegas, 2001, pp. 310-317.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

With the rapid growth of the Internet, especially World Wide Web, more and more information sources have become available online and heterogeneously distributed over the Internet. The need to search multiple collections in a distributed environment has been becoming an increasingly important problem commonly known as the resource discovery problem [SEK 92]. It is impractical to create a single centralized index that includes all the documents in all the collections. Information retrieval (IR) research, which traditionally studied centralized collections, faces new challenges in the distributed environments. As a result, distributed information retrieval has become an important area in IR research in recent years.

1.1 Distributed information retrieval

The need to search multiple collections in distributed environments is becoming increasingly important as the sizes of individual collections grow and network information services proliferate. Obviously, there is no sense in forwarding a user query to remote databases at each local site due to time requirement and the cost of transporting it all over the Internet. How to efficiently and effectively organize, manage and retrieve relevant information reactively and proactively in a dynamic and distributed environment is one of the most significant challenges faced by information retrieval research.

1.1.1 What is distributed information retrieval

A distributed information retrieval (DIR) system should be able to provide multiple users with a concurrent and efficient access to multiple text collections located on a remote site. So it typically consists of a set of server processes. Each runs on a separated processing node, and a designated broker process is responsible for accepting user requests, distributing the requests to the servers, collecting intermediate results from the servers, and combining the intermediate results into a final result for the user.

Generally speaking, a DIR system has the following *features*:

- To run each subtask on different computers and a network protocol is used to perform the communication between the subtasks.
- To select a subset of the distributed servers for processing a particular request rather than broadening all the servers to search.
- To search more documents.

1.1.2 Research issues in distributed information retrieval

To build an efficient and effective DIR system, there are some issues specific to distributed information retrieval which need to be considered. They are mainly composed of three fundamental activities:

- Information Source Selection (Resource Discovery or Collection Selection)

Information source selection is a procedure for the selecting a subset of the distributed collections, which are most likely to contain relevant documents for the current query

rather than broadcasting every request to every server in the system [fuhr 99, GGT 94].

- Query Processing

During a query processing, a query is distributed to the selected collection. Each of the participating search servers evaluates the query on the selected collections using its own local search algorithm, and produces a set of individual result-lists.

- Result Fusion (Result Merging or Collection Fusion)

Result fusion is a data fusion problem in which the results of retrieval run on separate and autonomous document collections must be merged to produce a single, effective result. It arises from incomparable ranking scores returned by searches of the different collections in a distributed environment. Directly merging results based on the incomparable scores hurts effectiveness [VGJ 95, YR 98].

1.2 Information source selection in distributed information retrieval

Obviously, it is infeasible and inefficient to make exhaustive searching of all collections in a realistic environment. To maintain effectiveness of distributed information retrieval, the system must be able to select the most relevant subset of collections in order to reduce the search space. The work on collection selection is directly beneficial to the execution performance of a distributed information system since searching fewer collections takes less time.

In recent works, a number of different approaches for database or collection selection have been proposed and individually evaluated so as to efficiently and effectively organize, represent and search distributed collections.

Xu and Croft [XC 99] proposed a cluster – based language model in which document clustering was used to organize collections around topics, and language modeling was used to properly represent topics and effectively select the right topics for a query.

Voorhees, et, al [VGJ 95] exploited the similarity of a new query to previously evaluated training queries and made use of relevant judgement from previous queries to compute the number of documents to retrieve from each collection.

Callan, et, al [CLC 95] presented that ranking collections could be addressed by an inference network in which the leaves presented document collections, and the representation nodes presented the terms that occured in the collection. The probabilities could be based upon statistics that were analogous to *tf* and *idf* in normal document retrieval, where *tf* and *idf* are always used to indicate the effectiveness of retrieval in information retrieval. The effectiveness of this approach was evaluated using the INQUERY retrieval system [CCH 92].

Gravana, et, al [GGT 94] used document frequent information of each individual collection to estimate the result size of a query in each collection and selected a set of most relevant collections with these estimates.

Fuhr [Fuhr 99] developed a decision – theoretic model and discussed different parameters for each database: expected retrieval quality, expected number of relevant documents in the database, and cost factors for query processing and document

delivery. He gave a divide – and – conquer algorithm to compute the overall optimum in order to receive the maximum number of documents at a minimum cost.

Yuwona and Lee [YL 97] described a centralized broker architecture in which the broker maintained *df* table for servers from the user query which can be best discriminated between servers, and then servers with higher *df* values for those terms were selected to process the query.

Other researches on information selection from multiple, distributed information sources have been studied under a variety of names, including Centralized Index [MZ 95], Broker Agents[DAN 91], Probabilistic Solution[Bau 97] and Server Selection[HT 99].

Frech, et, al [CP 2000, FPC 99, FPV 98] evaluated three of these approaches, CORI [CLC 95], CVV [YL 97] and GLOSS [GGT 94] in a common environment and found that there was a significant room for improvement in all approaches, especially when very few information sources were selected.

1.3 Motivations and contributions of this thesis

This thesis concentrates on information source selection in distributed information environments. The reason for selecting this topic is that the problem of information source selection is one of the fundamental problems in Distributed Information Retrieval (DIR) and it has not been solved satisfactorily.

Information can now be made available on the Internet very easily and at a very low cost with minimal effort. At the same time there exists some problems of using

the Internet and Web to retrieve information of interest to a user. These problems include:

- Inexact and ambiguous description of the user's query;
- Dynamic nature of the information on the internet;
- Distributed, heterogeneous nature of information and information services.

Although traditional IR techniques provide some effective algorithms to locate relevant documents from distributed information sources, they are "simple-minded" and are suffered from the following several deficiencies:

- Most of IR techniques use a strictly keyword-based search as opposed to a concept-based search.
- They are mostly lack in learning capability to improve the quality of its search results and adapt to the frequent changes of the dynamic environment.
- Most IR techniques can not make use of previous search experiences to help further limit the scope of information sources from inexhaustible pool of information for improving search effectiveness.

Intelligent agents, the products of an innovative technology, provide a promising solution to the problem of information overload on the Internet and the problem of information retrieval (IR). Due to exhibiting many key attributes such as *autonomy, intelligence* and *adaptability*, intelligent agents have potentiality of mitigating the complexity of information retrieval and management. It is better to combine intelligent agent with effective IR techniques to improve the performance of existed information systems [CM 2000].

In this thesis, we introduce a new approach, agent-based intelligent information source selection, which is an alternative way for overcoming the problems of information selection from distributed information sources. This approach makes use of three artificial intelligence techniques, including *query expansion* with *a Naive Bayes text classifier*, intelligent information selection with *case-based reasoning* and the improvement of the search quality and the adaptation to the dynamic web environment by learning the user's feedback with *reinforcement learning*. These three methods are put together to provide a guidance for designing an intelligent approach, which aims to optimally select potentially good information sources to search.

The major contributions in the thesis include:

(1) The system framework of the agent-based intelligent information source selection system is proposed. Such agent-based system compartmentalizes specialized task knowledge to different agents to mitigate the complexity of information source selection. The agent-based approach makes this system more scalable, flexible and extensible.

(2) A query expansion technique with *a Naive Bayes text classifier* for enriching the user's original query is presented. This technique is suggested to deal with the fundamental issue of word mismatch in information retrieval by adding related terms or concepts with similar meaning to those in the query.

(3) A *case-based reasoning* algorithm is used to select a set of promising databases to search. A query input case is used to find the best matching case in the case base which contains the information on what information source might be useful for the user's query based on the confident factor of every term in the query.

(4) An adaptation algorithm – *reinforcement learning* algorithm is developed to improve the quality of information source selection and to adapt to changing information source performance. The heuristic we use here is that the information sources' performance can be inferred from user feedback.

1.4 Structure of the thesis

The remainder of the thesis is organized as follows. In Chapter 2, some research work on intelligent agents for information retrieval on the Internet is firstly given. Then we propose a system framework for an agent-based intelligent information selection approach and briefly describe the function components used in this framework. In Chapter 3, we provide the details on the three artificial intelligent techniques described in Chapter 2, which are mainly used by three main components of this framework – Analysis Agent (AA), Case Matching Agent (CMA) and Learning Agent (LA). In addition, we have implemented the first component of this system - Analysis Agent and given some experimental results to support the effectiveness of query expansion technique with *a Naive Bayes text classifier.* Finally, we offer a summary of our contributions, and outline future work of this research in Chapter 4.

Chapter 2

A framework of an agent-based intelligent information source selection system

In the recent past, the field of software engineering has witnessed the emergence of agent based computing. There is a specific category of these agent systems which is so – called known as "intelligent agents" which embody techniques derived from the field of artificial intelligence (AI) learning, adaptation and user modeling. These intelligent agents can carry out some sets of operations with independence or autonomy by making decisions on the basis of data they acquires about the environment in which they find themselves, rather than as a result of direct instruction from the user.

2.1 Intelligent agents for information retrieval on the Internet

Due to the explosive growth of the Internet, finding specific information is becoming extremely difficult, sometime even frustrating for users or machine systems to collect, filter, retrieve, and use most relevant information in problem solving. The notion of intelligent agents has emerged to address this challenge [EW 94]. Intelligent agents are programs that act on behalf of their human users to perform laborious information-gathering tasks. These tasks include:

- Locating and accessing information from various on-line information sources.
- Resolving inconsistencies in the retrieved information.
- Filtering away irrelevant or unwanted information, integrating information from heterogeneous information sources, and adapting over time to human user's information needs.

Due to having the abilities of searching, retrieving, filtering and presenting relevant information, reactively and proactively, intelligent agents offer promising solutions to the current information explosion on the Internet and problems of information retrieval (IR). They have the potential to mitigate the complexity of information retrieval and management by providing a locus of intelligence. Agents could provide intelligent IR interfaces, or perform mediated searching and brokering, clustering and categorization, summarization and presentation. Agent based approaches make IR systems more scalable, flexible, extensible and interoperable.

These intelligent agents can perform certain tasks on behalf of their users in an autonomous fashion and with some level of pro-activity and/or reactivity. That can also exhibit some level of the key attributes of autonomy, intelligence and adaptability. These specific features help intelligent agents to act as an assistant to the user in carrying out the task of information retrieval.

There have been research activities at various governments, academic and industry research institutes to develop innovative agent – oriented technologies supporting public access to heterogeneous information sources distributed over the Internet [DK 97, YHM 98, DLP 95, Mul 94, SPW 96].

Das et al [DK 97] developed a scalable agent-based information retrieval engine named ***SAIRE***, which employed intelligent agents, natural language (NL) understanding, and conceptual search techniques to support Internet access to Earth and Space Science data at distributed technical centers across the US. ***SAIRE*** adopted a multi-agent architecture where various agents collaborated and communicated with each other to support intelligent information retrieval. Among those agents, two important agents *UMA* (User Modeling Agent) and *CSA* (Concept Search Agent) were introduced. These two agents worked together to derive a more exact query matching the user's expectation.

Yang et al. [YHM 98] developed intelligent mobile agents for customizable information retrieval from distributed data and knowledge sources. The *TFIDF* classifier was incorporated into mobile agents on the ***Voyager*** mobile agent platform to selectively retrieve documents from remote collections so that the saving of network connection in mobile agent systems would be even greater for the very large data.

MACRON [DLP 95] had an organizational architecture and used reasoning agents, low-level network retrieval agents, and user interface agents. The architecture consisted of functional and query-answering units, each of which was made up of a number of individual agents, including a facilitator. Functional units provided access to information sources, while query-answering units consisted of a query-manager agent and a set of agents selected from the set of functional units to process a given user's query.

Muller [Mul 94] proposed an intelligent multi-agent architecture for information retrieval on the Internet, which incorporated the idea of user modeling with machine learning methods into Web search services. This intelligent multi-agent system mainly consisted of 4 different types of agents, an interface agent, special wrapper agents, an integration agent and a user modeling agent, to perform different tasks.

Sycara et al. [SPW 96] developed a reusable, multi-agent computational infrastructure called *Retsina* (Reusable Task Structure-based Intelligent Network Agent). In the *Retsina* framework, each user was associated with a distributed collection of intelligent agents that ran across different machines and cooperated asynchronously to perform goal-directed information retrieval and integration for supporting a variety of decision-making tasks. *Retsina* mainly comprised three types of agents: interface, task and information agents.

Although working prototypes of several significant intelligent agents for information retrieval on the Internet are now in existence, agent based technique is still in its infancy, like the Web itself, and considerable research is necessary before such agents fulfil their potential capabilities.

2.2 A framework of an agent-based intelligent information source selection system

This section describes a framework of an agent-based intelligent information source selection system (***IISS***), our ongoing project, as shown in Figure 2.1. Directed arrow lines in the diagram represent data flow. This system consists of three main functional

components, namely, the *Case – Matching Agent* (CMA) component, the *Learning Agent* (LA) component and the *Analysis Agent* (AA) component. In addition, the system includes other two types of agents, a User Interface Agent (UIA) and Search Agents (SA_S).

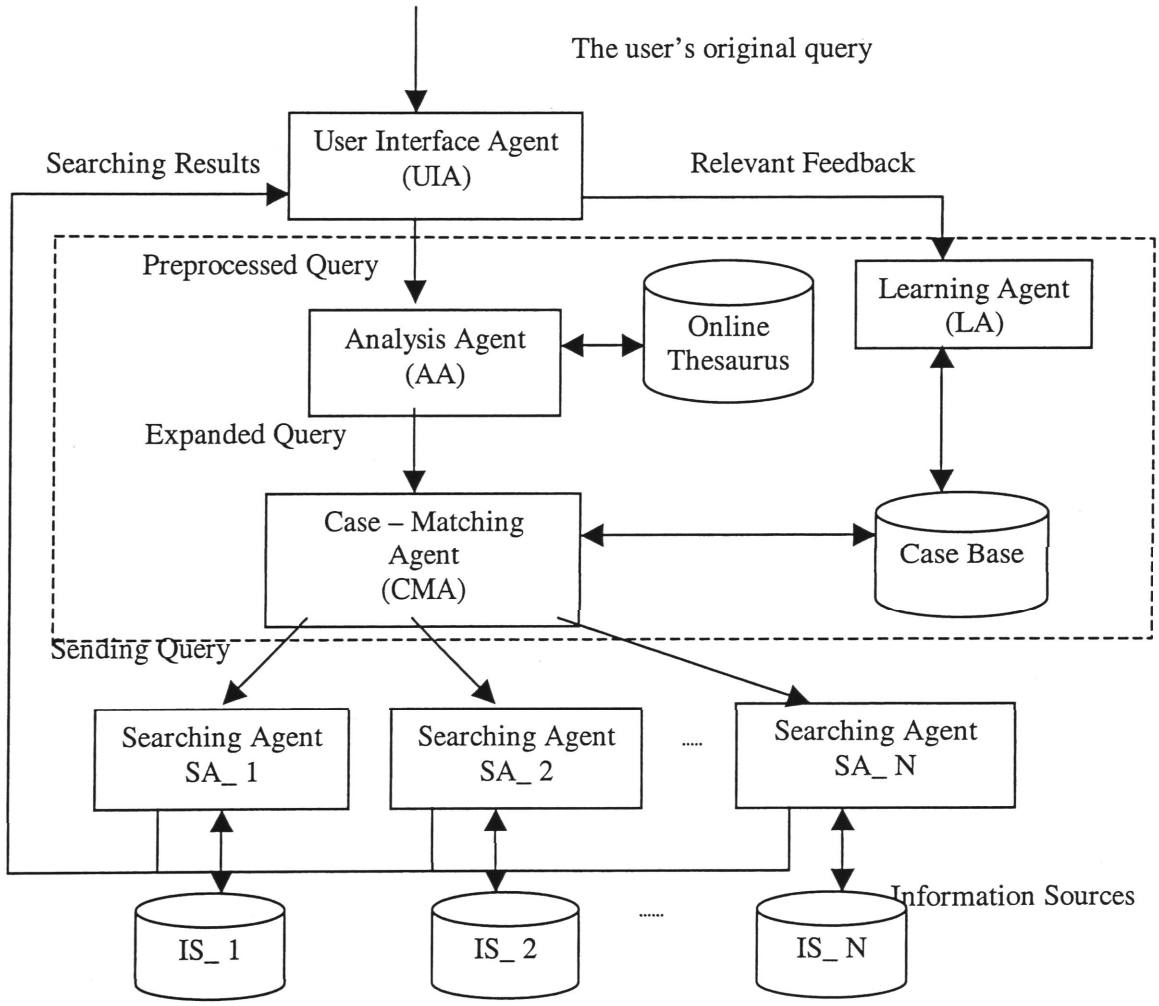

Figure 2.1: The framework of the agent-based intelligent information source selection system

The *User Interface Agent* (UIA) interacts with the user by receiving user queries and presenting relevant information, including searching results and explanations. Sometimes, it makes some simple preprocessing work of the user's initial query such

as removing words that are too frequent according to a stoplist and normalizing the words in the query, when necessary. In addition, it also observes the user's behavior and provides the *Learning Agent* with the information about the user's relevant feedback to searching results.

The *Analysis Agent* (AA) accepts the preprocessed query and takes the user's original query terms as representatives of the concepts in which the user is interested. It automatically expands the terms with a *Naive Bayes* classifier using a class hierarchy with a set of labeled documents in the online thesaurus, and adds other terms with similar meaning to those in the original query to enrich the representation of the user's query. The chances of matching words in relevant information sources are therefore increased.

The *Case – Matching Agent* (CMA) carries out the selection process on distributed information sources and is underpinned by *case-based reasoning*. The CMA can autonomously determine the most appropriate information sources to search using a case-based reasoning algorithm. A case base gives a hint of what information sources might be useful for the user's query based on the confident factors of information sources with respect to every term in the query. The information on "good" information sources deemed relevant by the CMA is transferred to the *Searching Agents* (SA_S) so as to seek for the associated information sources.

In ***IISS***, the *Searching Agents* (SA_S) act as wrappers and provide intelligent access to a heterogeneous collection of information sources. They can seek for and retrieve information required and process the information obtained. They aid in the tedious

task of retrieving the relevant information from distributed information sources. Finally, they pass on to the user only the information that the user is interested in.

The *Learning Agent* (LA) is responsible for keeping track of a user's relevant feedback via the *User Interface Agent* (UIA). By considering such feedback and using the statistical data stored in the case base, the LA can apply *reinforcement learning* algorithm to adjusting the values of confidence factors of matching cases in the case base and then restore the change into the case base.

2.3 Summary

Intelligent agents offer promising solutions to the current information explosion on the Internet and to the problem of distributed information retrieval due to exhibiting some key attributes such as autonomy, intelligence, and adaptability. In this chapter, we have proposed an agent-based approach for intelligent information source selection under a distributed information environment to overcome the limitations of traditional IR techniques. The framework of this approach is introduced and each component of the approach is concisely described to explain how these components iteratively work together to locate the most appropriate information sources to search. The following chapter will concentrate on the introduction of three major components in detail, which involve in three artificial intelligent techniques – *Naive Bayes learning, Case-based Reasoning* and *Reinforcement Learning.*

Chapter 3

Three major components of the framework

3.1 Analysis Agent

Most often, users have difficulties in formulating a request because they are unfamiliar with the contents of information sources, so their queries usually are very short. Such a short query tends to be inexact and ambiguous. To assist the user, the *Analysis Agent* (AA) attempts to provide a *conceptual retrieval* method, namely, *query expansion,* which can automatically expand the user's queries from an online thesaurus which stores word relationships. Such query expansion discovers related terms or concepts, along with their relationships with those in the user's query. Query expansion does not change the underlying information need, but makes the expanded query more suitable for information source selection.

In the AA, a query expansion method is provided by *Naive Bayes,* an established text classification algorithm (Lew 98, MN 98) based on Bayesian machine learning technique.

The online thesaurus is constructed by a class hierarchy with a set of labeled training documents. The class hierarchy contains a large number of classes organized into multiple levels such that classes at higher levels have broad meanings that those at lower levels. In general, a child class is more specific in meaning than its parent class. With such a class hierarchy, we can assign different concepts to appropriate classes in

the hierarchy. So, topic class hierarchies are an efficient way to organize and manage a large quantity of information that would otherwise be cumbersome. Knowledge about each topic class of interests is provided in the form of its title, and some most probable keywords, as calculated by *Naive Bayes* with a set of labeled training documents (see figure 3.1). The US Patent database, Yahoo and the Dewey Decimal System are all examples of topic hierarchies that exist to make information more manageable.

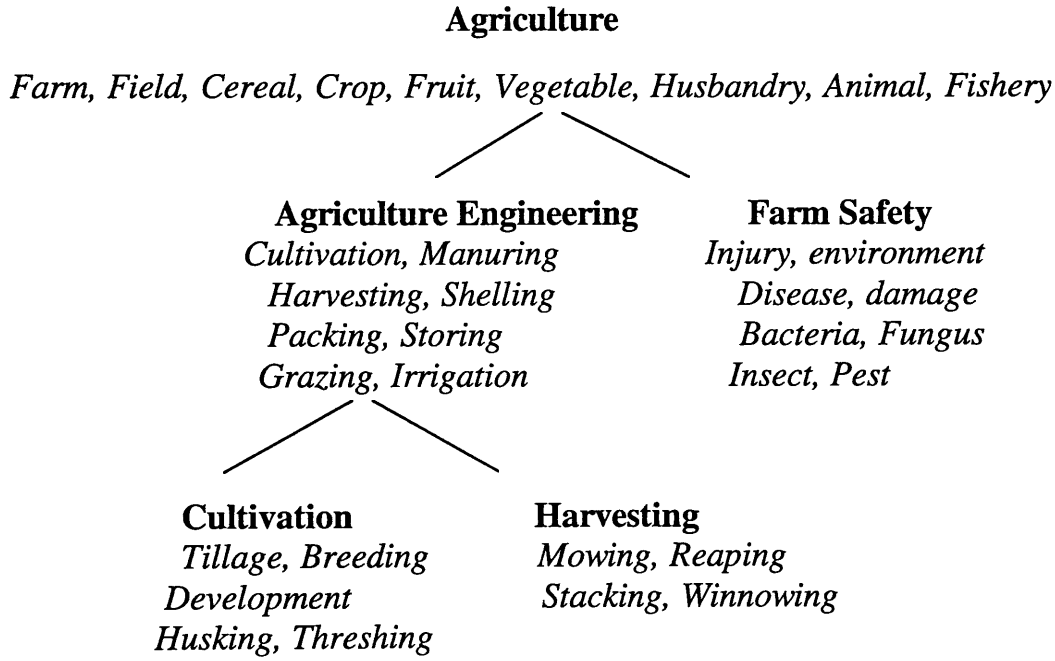

Figure 3.1: A subset of the topic hierarchy of online thesaurus

(Each node contains its title and the most probable keywords in italics calculated by *Naive Bayes* with training documents)

We consider a user's query to be associated with a *pseudo-document* indicated by *PD(Q)*. The content of the pseudo-document is a list of words with the weight that occurs in the preprocessed query, which can be defined as

$$PD(Q) = \{\langle t_i, w_i \rangle\} \quad (1)$$

where, t_i is terms (words) occurring in the user's query Q , and w_i is the term weight of the corresponding term in Q.

We then build an improved classifier by using the labeled training documents and the pseudo-document to bootstrap a Naive Bayes text classifier. This enhanced Naive Bayes classifier is used to discover new keywords that are probabilistically correlated with the original keywords in the pseudo-document.

These most probable keywords are ranked by the frequency that they occur in the training documents. Those top – ranking keywords from the same class as the pseudo-document *PD(Q)* will be added to the query and weighted appropriately. Terms in the original query are weighted more heavily than those terms which are not in the original query.

3.1.1 The *Naive Bayes* framework

We use the framework of multinominal *Naive Bayes* text classification (Lew 98, MN 98). The classifier parameterizes each class separately with a document frequency, and also word frequencies. Each class, c_j, has a document frequency relative to all other classes, written $P(c_k)$. For every word, w_t, in the vocabulary, V, $P(w_t|c_j)$ indicates the frequency that the classifier expects word w_t to occur in documents in class c_j.

Acquisition of these parameters is accomplished by using a set of labeled training documents, D. To estimate the word probability parameters $P(w_t|c_j)$, we count the frequency of a word w_t which occurs among all word occurrences for documents in class c_j. Then, the estimate of the probability of word w_t in class c_j is:

$$P(w_t|c_j)=\frac{1+\sum_{d_i\in D}N(w_t,d_i)P(c_j|d_i)}{|V|+\sum_{s=1}^{|V|}\sum_{d_i\in D}N(w_s,d_i)P(c_j|d_i)} \tag{2}$$

where, $N(w_t,d_i)$ is the number of times that word w_t occurs in document d_i; $P(c_j|d_i)\in\{0,1\}$, is given by the labeled training documents' class label; the vocabulary V, $V=\{w_1,w_2,\cdots,w_{|V|}\}$, $|V|$ is the number of all words occurring in documents in class c_j.

The estimate of the class prior parameters $P(c_j)$ is set in the same way:

$$P(c_j)=\frac{1+\sum_{d_i\in D}P(c_j|d_i)}{|C|+|D|} \tag{3}$$

where, $|C|$ indicates the number of classes, $c_j\in C=\{c_1,c_2,\cdots,c_{|C|}\}$; and $|D|$ is the number of the labeled training documents, $D=\{d_1,d_2,\cdots,d_{|D|}\}$.

3.1.2 Query expansion with a Naive Bayes classifier

Given an unlabeled document – a *pseudo-document PD(Q)* and a *Naive Bayes* classifier with the parameters $P(c_j)$ and $P(w_t|c_j)$ calculated from the labeled training documents, we can determine the probability that *PD(Q)* belongs to the class c_j using Bayes' rule and *Naive Bayes* assumption – that the probability of each word event in a document is independent of the word's context and position in the document.

$$P(c_j|d_{PD})=\frac{P(c_j)P(d_{PD}|c_j)}{P(d_{PD})} \tag{4}$$

where, (1) $P(d_{PD}|c_j)$ is the probability of a document given by its class:

$$P(d_{PD} \mid c_j) = \prod_{t}^{|d_{PD}|} (\delta + P(w_t \mid c_j)) \tag{5}$$

w_t is a word that occurs in a *pseudo-document PD(Q).* If w_t also occurs among all word occurrences for documents in class c_j, $P(w_t|c_j)$ can be got from Equation 2. But if w_t does not exist in class c_j, $P(w_t|c_j)$ is probably zero value. So we set a parameter δ, which is a very small value in the area (0,1) in order to avoid zero value in multiplication. (2) $P(d_i)$ is the probability of a document over all mixture classes *C*, $C = \{c_1, c_2, \cdots, c_{|C|}\}$.

$$P(d_{PD}) = \sum_{r=1}^{|C|} P(c_r) P(d_{PD}|c_r) \tag{6}$$

So, we can calculate the posterior probability of each class given the evidence of the unlabeled document *PD(Q).*

$$P(c_j|d_{PD}) = \frac{P(c_j)P(d_{PD}|c_j)}{P(d_{PD})} = \frac{P(c_j)P(d_{PD}|c_j)}{\sum_{r=1}^{|C|} P(c_r)P(d_{PD}|c_r)}$$

$$= \frac{P(c_j)\prod_{t=1}^{|d_{PD}|}(\delta + P(w_t|c_j))}{\sum_{r=1}^{|C|}\left(P(c_r)\prod_{t=1}^{|d_{PD}|}(\delta + P(w_i|c_r))\right)} \tag{7}$$

Finally, we select the class with the highest probability that most probably contains the words with similar meaning to those in a query.

Some keywords with the higher value of $P(w_t|c_j)$ in the same class as the *PD(Q)* are chosen to expand the user's query, but the weight of those expanded terms (keywords) in the query will be downweighted by reducing the weight of original query terms.

3.2 Case – Matching Agent

In the Case – Matching Agent (CMA), expanded query is actually evaluated with a case – based reasoning algorithm (Zhang 98) to select a set of promising information sources to search.

The CMA contains an information source index, which consists of a set of virtual documents. A *virtual document (VD)* is a list of words (terms) and their confidence factors in the corresponding information source. More formally, the virtual document for an information source *IS* is:

$$VD(IS) = \{\langle t_i, CF_i \rangle\} \tag{8}$$

where t_i is a term (word) occurring in *IS*, $i = 1,2,\cdots,$ and $CF_i \in (0,1)$, is the confidence factor that a certain *IS* satisfies the information need expressed by t_i; CF_i can be achieved by $TF \times IDF$ method based on *Vector Space Model* [SM 83], which is described as follow:

$$CF_i = \sum_{d_k \in IS} tf_k \cdot \log N/df \tag{9}$$

where, tf_k is the term frequency for a term t_i in document d_k and df is the number of documents in the collection c_i of N documents in which term t_i occurs.

Once the CMA receives an expanded query, it produces a query input case, and then interacts with the case base to find the best matching case so as to determine the most appropriate information sources to search.

3.2.1 Definitions and case representations

The basic idea of case – based reasoning (CBR) is to solve a new problem (an input case) by reusing solutions that were used to solve old problems (existing cases in the case base) [RS 1989]. CBR can reason in depth about a problem case and in particular, retrieve highly relevant cases, but this ability is limited by the availability of cases actually represented in the case base. On the other hand, full-text information retrieval systems are not hampered by any lack of available cases (in textual form) but they cannot reason about a problem case and their sense of relevance is very weak.

A natural approach is to form a hybrid system with CBR and traditional information retrieval techniques where the strengths of each are used to overcome the weakness of the other in order to produce results or functionality unachievable by either individually. Our goal in this project is to take advantage of both the highly articulated sense of relevance used in CBR and the broadly applicable retrieval techniques in IR in order to retrieve the most appropriate collections to search.

Our hybrid CBR-IR approach takes a query input as a standard frame-based representation of a problem case and outputs a set of relevant collections retrieved from distributed information sources.

To better explain the CBR-IR strategy, we first make the following definitions.

Definition 1: A query input is the matrix representation of multiple results from different information sources (see Matrix 1).

$$\begin{bmatrix} CF_{11} & CF_{12} & \cdots & CF_{1n} & W_1 \\ CF_{21} & CF_{22} & \cdots & CF_{2n} & W_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ CF_{m1} & CF_{m2} & \cdots & CF_{mn} & W_m \end{bmatrix} \quad \text{(Matrix 1)}$$

where CF_{ij} ($1 \le i \le m$, $1 \le j \le n$) represents the confidence factor of ith term in the user's query in jth information source IS_j; W_i ($1 \le i \le m$) is the weight of ith term in the user's query (see Function 1 in Section 3.1); m indicates that there are m terms in the user's query and n is the number of distributed information sources. If ith term does not appear in IS_j, $CF_{ij} = 0$ (see Function 8).

Definition 2: An output is the final result merging of the vector $(CF_{*1} \quad CF_{*2} \quad \cdots \quad CF_{*m})$ after the merging of the input matrix 1. * indicates the merging result from corresponding values with the subscription of 1, 2, ···, n in the same place.

Definition 3: An existing case in the case base consists of an input matrix and an output vector.

Examples of both an input case and an existing case in the case base are shown in Table 3.1 and Table 3.2, respectively.

	Query Terms	IS 1	IS 2	IS 3	Weight
Multiple Results	Term1	0.5	0.1	0.2	1
	Term2	0.2	0.3	0.8	0.5
	Term3	0.7	0.2	0.5	1

Table 3.1: An example of a query input case

	Query Terms	IS 1	IS 2	IS 3	Weight
Multiple Results	Term1	0.4	0.2	0.3	1
	Term2	0.2	0.1	0.8	0.5
	Term3	0.8	0.2	0.4	1
Final Result Merging	Query Terms	1.4	0.45	0.94	

Table 3.2: An example of an existing case in the case base

Firstly let's examine an example. There are three information sources (e.g. IS_1, IS_2, IS_3) to search for a expanded query Q which consists of three terms, *Term 1, Term 2* and *Term 3*. *Term1* and T*erm 3* belong to the original query, while *Term 2* comes from query expansion with a *Naive Bayes* classifier. So they have different weights, separately, 1 and 0.5.

When the CMA receives this expanded query, it firstly produces automatically a query input (see Table 3.1). Then it searches the case base to find the best matching case. Finally, it gets a suitable matching case – *Case A* (see table 3.2) by case-based reasoning (the detail will be described in section 3.2.2 below). Judging from the output vector $(CF_{*1}\ \ CF_{*2}\ \cdots\ CF_{*n})$ in *Case A*, it regards IS_1 and IS_3 as the appropriate information sources to search for the query Q.

3.2.2 The principle of a CBR-IR approach

The basic idea of the case – based reasoning is to solve a new problem (an input case) by reusing solutions that were used to solve old problems (existing cases in the case base). Figure 3.2 demonstrates the principle of our CBR-IR approach.

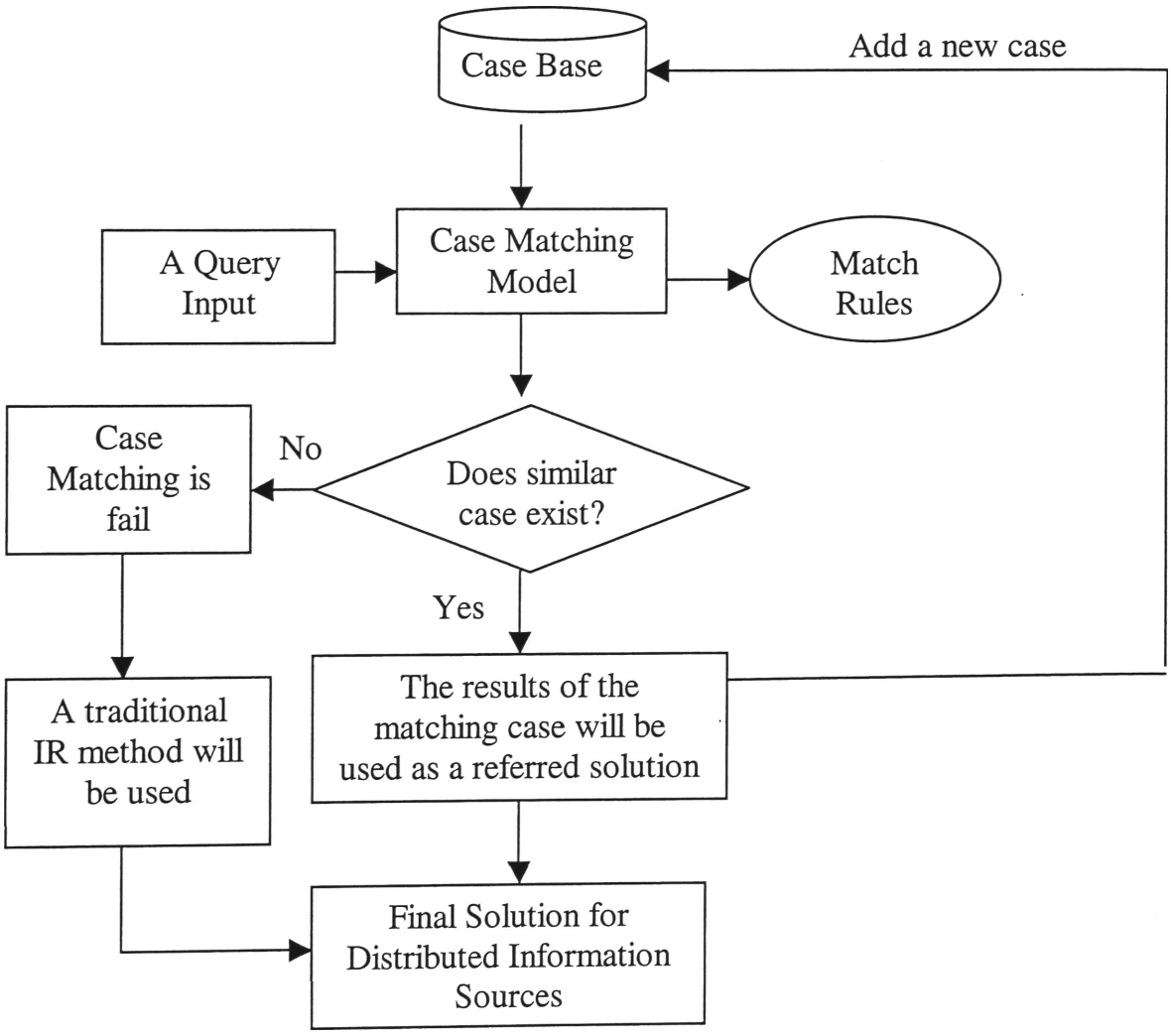

Figure 3.2: The principle of the CBR-IR approach

For a query input case, the CMA will search the case base to find the best matching case. Once a suitable case is found, its corresponding result might be used as a referred

solution for the input case. The referred result will be considered as a new case that might be added to the case base. If there is no similar case available in the case base, the matching procedure is not successful.

Normally, it is not easy to get a perfect matching for a query input. After searching the case base, the CMA advises the nearest matching case which most nearly solves the selection problem.

In order to increase the CMA's efficiency, it is necessary to classify the searching space. There are six case matching rules in the matching rule set which are implemented to classify the searching space in different levels. The classification of case matching is shown in Table 3.3

Error	Level	Performance	Matching Result
5%	L1	Choose the solution from the matching case	Good
5% -10%	L2	Choose the solution from the matching case	Good -
10% -15%	L3	Choose the solution from the matching case	Acceptance +
15% -20%	L4	Choose the solution from the matching case	Acceptance
20%- 25%	L5	Choose the reference from the matching case	Acceptance -
>25%	L6	There are no solution for the case base	Fail

Table 3.3: Case matching rules

Definition 4: The mean error $\overline{error}$ for a case matching is defined as:

$$\overline{error} = \frac{\sum_{i=m}\left(\sum_{j=n}\left|CF_{ij}\cdot W_i - CF'_{ij}\cdot W'_i\right|\right)}{m*n} \qquad (10)$$

where, CF_{ij} represents the confidence factor of *i*th term in the user's query from the information source IS_j for the input case; CF'_{ij} represents the confidence factor of *i*th term in the user's query from the information source IS_j for an existing case; $W_i (1 \le i \le m)$ is the weight of *i*th term in the user's query; $W_i' (1 \le i \le m)$ is the weight of *i*th term in the existing case; *m* indicates that there are *m* terms in the user's query and *n* is the number of distributed information sources (see Matrix 1).

Definition 5: The maximum error $error_{\max}$ for a case matching is defined as

$$error_{\max} = \max\left\{\left|CF_{ij} \cdot W_i - CF'_{ij} \cdot W_i'\right|\right\} \tag{11}$$

where $i = 1,2,\cdots,m$, $j = 1,2,\cdots,n$. If for a case matching, $error_{\max} = 0$, we call this matching a perfect matching.

The CMA starts to search the case base from level 1 (L1 in Table 3). If a matching case is not found in level 1, the searching area will be extended to level 2 (L2), and so on until a similar case is found or the whole searching space has been examined.

The final result merging vector $(CF_{*1} \quad CF_{*2} \quad \cdots \quad CF_{*n})$ for an input case will depend on the level which the match results fall in. For *j*th information source, the final confidence factor CF_{*j} $(1 \le j \le n)$ can be calculated as:

$$CF_{*j} = \begin{cases} CF'_{*j} & level \in (L_1, L_2, L_3) \\ CF'_{*j} \cdot \left(1 - \overline{error}\right) + \left(\sum_{i=1}^{m} CF_{ij} \cdot W_i\right) \cdot \overline{error} & level \in (L_4, L_5) \\ \sum_{i=1}^{m} CF_{ij} \cdot W_i & level \in (L_6) \end{cases} \tag{12}$$

Where CF'_{*_j} represents the confidence factor of the user's query for an existing case, which can be gotten by $\sum_{i=1}^{m} CF'_{ij} \cdot W'_i$.

When the matching result falls in the area of L_3 or higher levels, we consider that the CMA can find an appropriate case whose corresponding result in the existing case might be used as a referred solution for the input case. If the matching result is a little unsatisfactory which falls in the area of L4 or L5, we need to recalculate the output vector with the values of the mean error $\overline{error}$, the confidence factors of the most roughly matching case and those of the input case. But we still pay more attention on the result of the most roughly matching case for the final output. Under the two above situations, the referred result together with the input case will be considered as a new case that might be added to the case base. If the matching is a completely failure, we have to use a traditional IR method that is to make use of the confidence factors in the input case to get the output vector rather than the existing case in the case base.

Finally, the CMA chooses some top-rank information sources with higher CF_* from the final result of the output vector as potential "good" information sources to search.

3.3 Learning Agent

In complex realistic environments, we usually cannot get a perfect existing case from the case base to match a query input case. If such suitable matching case does not exist, the CMA will only advise a roughly closest matching case to the user. Both the final rough solution and the current input case might be considered as a new case and

might be added to the case base for further case matching. Moreover, ***IISS*** originally uses a hand – built, relatively static case base, which does not reflect the dynamic changing performance of each of the underlying information sources.

As a result, it is necessary for the system to have learning capability to improve the quality of its search results and adapt to the frequent changes of the dynamic environment. User feedback is always considered as a useful indicator, which assesses the effectiveness and efficiency of the search result and reflects the latent change of information sources under the dynamic environment. The learning agent (LA) learns user feedback and continuously updates confidence factors of matching cases in the case base for the quality of selection using an adaptation algorithm – reinforcement learning algorithm.

In order to explain the principle of learning agent, it is necessary to briefly describe what is a reinforcement learning model in the following subsection.

3.3.1 A reinforcement learning model

In machine learning, the term "reinforcement learning" refers to a framework for learning optimal decision making from rewards or punishment [KL 96]. Upon taking an action, the learner is simply told how good or bad the selected action is, expressed in the form of a scalar "rewards".

Problems with reinforcement learning are well modeled as *Markov Decision Process* (MDP) [SPW 98, KL 96]. A MDP consists of a set of states, $s \in S$, a set of actions, $a \in A$, a reward function R: $S \times A \rightarrow R$, which specifies expected instantaneous reward as a function of the current state and action, and a state –

transition function T: $S \times A \rightarrow S'$. *T(s, a, s')* is the probability of making a translation from state *s* to state *s'* taking action *a*. The goal of reinforcement learning is to learn a policy, a mapping from perceived states *S* of the environment to actions *A* to be taken when in those states, $\pi : S \rightarrow A$, that maximizes the sum of its reward over time.

Recall that a policy, π, is a mapping from states, $s \in S$, and actions, $a \in A$, to the probability $\pi(s,a)$ of taking action *a* when in state *s*. Informally, the *value* of a state *s* under a policy π, denoted $V^{\pi}(s)$, is the expected return when starting in *s* and following π thereafter. For MDP, $V^{\pi}(s)$ can be defined as:

$$V^{\pi}(s) = \sum_{t=0}^{T} E\left(\gamma^t r_t\right) \tag{13}$$

where $E_{\pi}\{\ \}$ denotes the expected value given that the agent follows the policy π, and r_t is the reward received *t* time steps after starting in state *s*. γ, $0 \leq \gamma \leq 1$, is a discount factor which makes sooner rewards more valuable than later rewards; the function $V^{\pi}(s)$ is called the *state – value* function for policy π.

Similarly, the value of taking action *a* in state *s* under a policy π, is denoted by

$$Q^{\pi}(s,a) = E_{\pi}\left\{\sum_{t=0}^{T} \gamma^t r_t\right\} \tag{14}$$

Q^{π} is called the *action – value function for policy π.*

The optimal policy, written π^*, is the one that maximizes the value, for all state *s*.

$$V^*(s) = \max_{\pi} V^{\pi}(s) = \max_{\pi} E\left(\sum_{t=0}^{T} \gamma^t r_t\right) \tag{15}$$

This optimal value function is unique and can be defined as the solution to the simultaneous equations which assert that the value of a state s is the expected instantaneous reward plus the expected discounted value of the next state, using the best available actions.

$$V^*(s) = \max_{\pi}\left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s'), \forall s \in S \right) \quad (16)$$

The optimal policy in terms of Q is defined by selecting from each state the action with the higher expected future reward:

$$\pi^*(s) = \arg\max_a Q^*(s,a) \quad (17)$$

Given an optimal polity π^*, the optimal value of each state, $V^*(s)$, for $s \in S(s_1, s_2, \cdots, s_n)$ can be straightforwardly calculated by dynamic programming called *value iteration* algorithm that can be shown to converge to the correct π^* values [KL 96].

Input π, the policy to be evaluated
Initialize $V(s) = 0$, for all $s \in S$
For each $s \in S$
For each $a \in A$
$Q(s,a) \leftarrow R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s')$
$V(s) \leftarrow \max_a Q(s,a)$
Output $V(s)$

Figure 3.3: Value iteration algorithm

3.3.2 Reinforcement learning for user feedback

In order to practically apply reinforcement learning to information selection, the heuristic we use here is that the information sources' performance can be inferred from the user's feedback. One of the fundamental features of the search result page returned by ***IISS*** system is the page summaries generated using local context around the query terms, which allows the users to more readily determine if the document answers his or her specific query. A user therefore finds documents of high relevance by quickly scanning the local context of the query term. So we can suppose that the earlier a document's link on the result page is chosen by the user, the better the information source contributing to this document performed on the query than those information sources contributing to other documents chosen later or even ignored by the user.

Now we need to make some simplifying assumptions in order to make the problem tractable and to aid generalization. The assumptions we choose initially are the following *six*:

1. Suppose that there are only N information sources chosen by the CMA to search for the user's query.
2. During the process of user browsing the result page, the action a is following a particular document's hyperlink in the result page.
3. The state s is an information source contributing to the document chosen by the user.
4. A policy π is the behavior that the user browses the result page and chooses the documents that he/she is interested in.

5. The reward value of an information source $R(s,a)$ can be gotten from the output vector $(CF_{*1} \quad CF_{*2} \quad \cdots \quad CF_{*m})$.
6. The value of the state-transition function *T(s, a, s')* is the same for all $s \in S$ which is *1/N,* since the chance that each time the user click the hyperlink of document which any information source probably contributes to is same.

So we can get the optimal value $V^*(s)$ by *Value iteration algorithm* (see Figure 3.3), thus learning an optimal policy π^*. The optimal policy π^* is the one that maximizes the value $Q(s,a)$. According to the order of information source contributing to the documents chosen by the user in the optimal policy π^*, we can correspondingly adjust the confidence factors of the existing matching case contributing to the final merging result for the formation of the result page. The earlier an information source appears in the optimal policy π^*, the larger CF_* of such information source will be.

To explicitly explain the principle of reinforcement learning used for the information source selection, let us give an example. The existing case in Table 3.2 can be turned into a simple example of reinforcement learning by simplifying it and providing some details (Our aim is to produce a simple example, not a particularly realistic one). Suppose that there are 4 different users who use the same query during the period of reinforcement learning. Each user chooses 6 documents on average to visit from the result page. These documents come from 3 different information sources, separately, IS_1, IS_2, IS_3. The orders that the 4 users browse the documents are shown in Table 3.4.

	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
User 1	IS_1	IS_3	IS_1	IS_1	IS_2	IS_2
User 2	IS_2	IS_1	IS_3	IS_3	IS_2	IS_3
User 3	IS_1	IS_1	IS_3	IS_2	IS_1	IS_2
User 4	IS_3	IS_1	IS_1	IS_2	IS_2	IS_3

Table 3.4: The orders of information sources that contribute to the documents the users browse

Then we hope to learn an optimal policy, π^*, that maximizes the sum of its rewards over time. For each information source, the value of the reward $R(s,a)$ is known from Table 3.2. We suppose that γ, the discount factor, is 0.2, and *T(s, a, s')* is *1/3*. Therefore, we can easily get the values of Q^π by Equation 14, which is $\{Q_1^\pi,\ Q_2^\pi,\ Q_3^\pi,\ Q_4^\pi\} = \{1.484,\ 0.578,\ 1.528,\ 1.276\}$. Then the optimal policy will be π_3, which is the order that User 3 chooses the documents from the result page. According to the order and the time that different information sources appear in π_3, we will weigh the confidence factor CF_{*1} heavier than other two confidence factors CF_{*3} and CF_{*2}.

3.4 The implementation of Analysis Agent

IISS, an agent –based intelligent information source selection system, is our ongoing project. Currently, we have finished the implementation of the first major component

– *Analysis Agent (AA)*, other two main components, *Case Matching Agent (CMA)* and *Learning Agent (LA)*, will be realized in future work.

In the experiment reports here, we only examine selection and retrieval performance in distributed environments using query expansion technique with a *Naive Bayes* classifier. The experiments are carried out on the testbed of the **Reuters 21578 Distribution 1.0** data set. The collections in this data set are indexed separately to simulate a real-world distributed IR system.

Query document and collection description are *automatically generated*: Queries and documents are indexed by eliminating stop words and then applying Porter's stemming algorithm [Por, 80]. The distributed IR system used in this experiment adopts a widely used technique: it creates a collection selection index. The *collection selection index* consists of a set of virtual documents, each of which is a light-weight representation of a collection. Specifically, the virtual document for a collection is simply a complete list of words in that collection and their weight are calculated by the famous formula $tf \cdot idf$ [SM 83].

To clarify the retrieval performance of searching a set of distributed collections using query expansion with *Naive Bayes* learning, we firstly discussed a general framework in our experiment which is:

1. To expand the user's query using *Naive Bayes* learning with m expanded concepts in the same class as the query, and then to add those concepts to the original query with decreasing weights.
2. To run the expanded query against the collection selection index and to get the n highest-ranked collections.

3. To search each of the n top ranked collections, and to return a list of the k top ranked documents from each collection; to merge the returned results, and then to evaluate the quality of the merged list of documents.

3.4.1 Experimental setup

Under this general framework, we will consider a number of variations and evaluate the impact that these variations had on the final document retrieval results. These variations are:

- Query expansion vs without query expansion for collection selection, and for document retrieval.
- The effect of adding different number expansion concepts on retrieval effectiveness.
- The effect of varying the size of the labeled training collection on retrieval effectiveness.
- The effect of assigning different weights to the expanded concepts on retrieval effectiveness.
- The effect of increasing the number of collections selected on retrieval effectiveness

We planned experiments with these variations, and evaluated the impact that these variations had on the collection selection and on the final document retrieval result. Descriptions of the testbed, details of the selection and merging approaches, and a more detailed description of the evaluation approaches are given below.

3.4.1.1 Testbed

In our experiments, we proposed a test environment for the systematic study of distributed information retrieval algorithms. Our testbed was based on the **Reuters 21578 Distribution 1.0** data set that consisted of 21578 articles and 135 topic categories from the Reuters newswire [Lew, 97].

When a query is posed, the system first expands it by *Naive Bayes* learning on the training data set and then searches for it on the actual set of distributed collections. Ideally, we would like the documents in the training collections and those in the actual collections to have similar coverage of subject matters in order to expand the query properly. So, we decomposed the **Reuters 21578** data set into two subdata sets – *REUTER-TEST* used for distributed collection and *REUTER-TRAINING* which was solely for the purpose of query expansion.

Sets of collections	REUTER-TRAINING	REUTER-TEST
Number of queries	96	96
Raw text size in megabytes	9.68	70.5
Number of documents	3294	17309
Mean words per document	176	176
Mean relevant documents per query	33	168
Number of words	5808	29568
Number of collections	96	200
Mean documents per collections	33	100

Table 3.5: Statistics about the sets of collections used for evaluation

General characteristics of these two subdata sets and the query sets appear in Table 3.5. In *REUTER-TEST*, we partitioned the large collections into 200 smaller collections of roughly equal size (about 100 documents each) that serve as hypothetical "information sources" in our distributed information retrieval test

environment. Each collection contains documents of several different topic categories. Each topic category has roughly 15 corresponding collections that contain relevant topic documents.

To guarantee enough labeled training documents for *Naive Bayes* learning, we chose 96 populous class documents from 135 topic categories as the training collections of the *REUTER-TRAINING* data set. Each collection only contains relevant documents of one topic class so as to acquire some most probable keywords about such topic class, which are calculated by *Naive Bayes* learning with these labeled training documents.

The **Reuters 21578** data includes a set of fielded topics, each of which is a statement of information need. We used 96 populous topics in the *REUTER-TRAINING* data set to construct a set of short queries. They can undergo automatic query expansion by *Naive Bayes* classifiers to construct a corresponding set of longer expanded queries.

3.4.1.2 Evaluation – baselines for comparison

Two baselines are referred to in the evaluation below, specifically:

(1) One is the optimal relevance-based ranking O_q for a single query q, which is used for evaluating the collection selection performance. The ranking order is produced by processing each query at each of the 200 test collections in the *REUTER-TEST* data set and then using the weight (see Equation 18 below) to rank the test collections. The algorithm for ranking test collections for a single query q is similar to the well-known

$tf \cdot idf$ approach [SM 83] by replacing tf (the term frequency in one document) with df and idf (inverse document frequency) with icf. It is defined as follows:

$$Weight(q \mid c_i) = df \times icf \tag{18}$$

where df is the document frequency of documents in a certain collection c_i of the *REUTER-TEST* data set. Those documents are those that belong to the same topic class as the query q; icf, inverse collection frequency, can be calculated as $\log(N/cf)$. N is the number of all collections in the *REUTER-TEST* data set, and cf is the number of collections in the *REUTER-TEST* data set which contain the same topic class documents as the query.

O_q is a ranking order where the collection with the largest weight is ranked 1, the collection with second largest weight is ranked 2, and so on.

(2) The other baseline is the retrieval performance of searching a set of distributed collections using the basic queries without query expansion. Comparison with this baseline tells us the improvement we have made by using *Naive Bayes* learning technique to expand the user's query.

3.4.2 Query expansion for collection selection

3.4.2.1 Evaluation methodology

The mean-squared root error metric was used to compare the effectiveness of variations to the basic collection ranking algorithms. The mean-squared root error of the collection ranking for a single query is calculated as:

$$\frac{1}{|C|}\cdot\sqrt{\sum_{i\in C}(O_i - R_i)^2} \tag{19}$$

where: (1) O_i is optimal rank for collection i, based on the weight scorn of relevant documents it contained (see section 3.4.1.2); (2) R_i is the rank for collection i determined by the retrieval algorithm, which is described in the following:

$$R(q|C_i) = \sum_{t_j\in q}\sum_{d_k\in C_i} tf_{jk}\cdot\log N/df_{t_j} \tag{20}$$

where $R(q|C_i)$ is the relevant score of the query q in the collection c_i; tf_{jk} is the term frequency for a term T_j of the query q in document d_k and df_{t_j} is the number of documents in the collection c_i of N documents in which term T_j occurs. The collection with the largest value of $R(q|C_i)$ is ranked 1, the collection with second largest value is ranked 2, and so on; (3) C is the set of collections being ranked.

The mean-squared root error metric has the advantage that it is easy to understand (an optimal result is 0), and it does not require labeling a collection 'relevant' or 'not relevant' for a particular query.

3.4.2.2 Selection result

Although we have argued that ranking collections is analogous to ranking documents (see Section 3.4.1.2), there are still some differences. The reason for ranking collections is *not* to find collections about a particular subject; it is to find collections containing as many documents as possible about the subject.

We first report the results of using query expansion in the collection selection stage only. As we expected, query expansion with *Naive Bayes* learning does improve collection selection. Experimental results on the REUTER-TEST data set support this, as shown by Table 3.6 and Figure 3.4. The mean-squared root errors for query expansion, averaged over 96 queries, are noticeably smaller than that for the base query.

There are a number of interesting things to observe in Table 3.6 and Figure 3.4. Firstly, when more collections are selected for searching, mean-squared root error tends to greater. This is understandable that while selecting more collections increases the chance of selecting a relevant-rich collection. It is not guaranteed to select collections in a right order. Secondly, the greatest improvement can be seen when 30 expanded concepts are used for selection (instead of 50 concepts). For these queries, expanding more concepts does not provide a large benefit. This may be due to 30 expanded concepts which contain most relevant concepts in term to original query. Expanding more concepts may not improve performance, sometime even degrade it.

REUTER-TEST 200-collections Testbed (96 queries)				
Mean-Squared Root Error at *s* collections selected	**50 Concepts**	**30 Concepts**	**10 Concepts**	**Base Query**
20 Collections	0.4847	0.4667	0.4901	0.5056
15 Collections	0.3364	0.3256	0.3347	0.3523
10 Collections	0.2763	0.2667	0.2836	0.3042
8 Collections	0.2515	0.2467	0.268	0.2923
5 Collections	0.2087	0.2016	0.224	0.2436
2 Collections	0.1423	0.1196	0.168	0.2145

Table3.6: The effect on mean-squared root error of varying query expansion size for selection performance on the REUTER-TEST

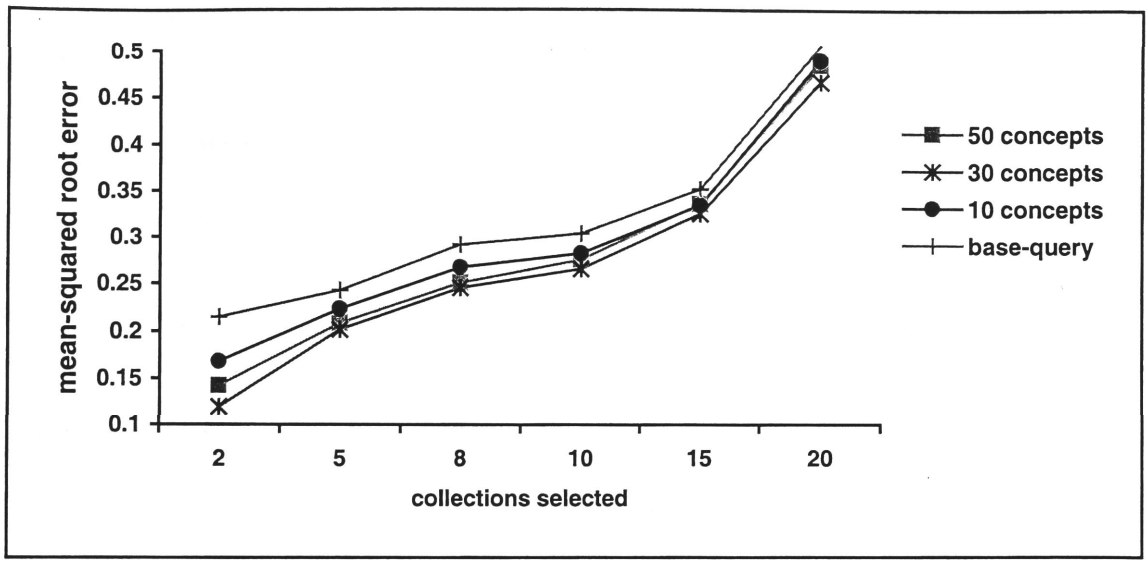

Figure 3.4: The effect of expansion concept size on selection performance in the REUTER-TEST

3.4.3 Query expansion for collection selection and retrieval

Although the improvement on collection selection is significant, we believe that the results do not reflect the power of query expansion for information retrieval. So we still expect that retrieval performance will be better than that of using the base query in both the collection selection and the retrieval stages.

Two common measures of retrieval effectiveness are *recall* and *precision* [SM 83]. But in a realistic environment, precision at low recall is far more important, because a typical user can only afford searching a small number of documents. We only search the top 10 collections in the estimated ranking ordered by $R(q|C_i)$ for a query, and retrieve a maximum of 50 top rated documents from each collection and merge them according to their relevant weights that are calculated by the famous formula $tf \cdot idf$ [SM 83]. In order to be able to investigate the retrieval effectiveness with query

expansion, we measure the precision of the first *s* top ranked documents ranging from 10 to 100.

The goal of the experiments in this section is to confirm the effect of a number of variations concerning the benefit of query expansion on distributed information retrieval (see section 3.4.1).

3.4.3.1 Expansion concepts

First, we compare different query expansion sizes with *Naive Bayes* learning and base query in term of retrieval effectiveness. It is interesting to see how the number of expansion concepts used affects retrieval performance. To see it more clearly, we plot the performance curve in Figure 3.5. Table 3.7 and Figure 3.5 show the effect of query expansion size on retrieval performance on REUTER-TEST compared to the retrieval baseline of base query.

Experiment results show that query expansion does improve retrieval performance if the number of expansion concepts is chosen properly. Reducing the number of concepts from 50 to 30 does not apparently affect retrieval effectiveness. In fact, using 30 concepts is even slightly better than using 50 concepts. But when only 10 concepts are used per query, retrieval performance suffers, by 9.82% on average. One possible problem is that query expansion with only 10 concepts cannot provide some so-called topic words which by themselves are very strong indicators of relevance. So those non-relevant expansion concepts hurt retrieval effectiveness.

Analysis of the results also reveals that for more than 30 expansion concepts, retrieval is improved at all documents cut-offs. Improvement at higher cut-offs (from 70 to 100) is around 7%, which is more noticeable than at lower cut-offs.

REUTER-TEST 200-collection testbed				
Precision at *s* docs (% change)	**Concepts 50**	**Concepts 30**	**Concepts 10**	**Base - Query**
10	0.4800 (+1.05)	0.4800 (+1.05)	0.4450 (-6.32)	0.4750
20	0.4625 (-0.47)	0.4750 (+2.21)	0.4225 (-9.08)	0.4647
30	0.4699 (+3.84)	0.4666 (+3.11)	0.4099 (-9.41)	0.4525
40	0.4562 (+0.48)	0.4612 (+1.58)	0.4000 (-11.89)	0.4540
50	0.4550 (+0.22)	0.4600 (+1.32)	0.4011 (-11.65)	0.4540
60	0.4533 (+2.88)	0.4574 (+3.81)	0.3990 (-9.44)	0.4406
70	0.4528 (+5.49)	0.4574 (+6.57)	0.3864 (-9.97)	0.4292
80	0.4601 (+7.63)	0.4535 (+6.08)	0.3831 (-10.38)	0.4275
90	0.4422 (+4.46)	0.4450 (+5.12)	0.3772 (-10.89)	0.4233
100	0.4355 (+5.32)	0.4450 (+7.62)	0.3755 (-9.18)	0.4135

Table3.7: The effect of query expansion size on retrieval performance in the REUTER-TEST

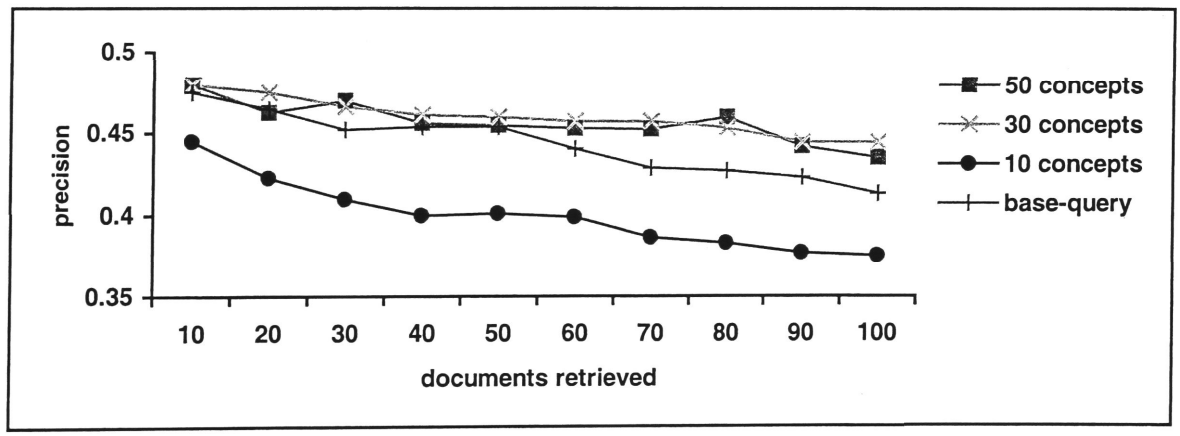

Figure 3.5: The effect of query expansion size on retrieval performance in the REUTER-TEST

3.4.3.2 Selection Collections

We are also interested in the impact of selecting more or fewer collections to search on retrieval performance. In general case, selecting more collections increases the chances of selecting a relevant-rich collection with the most (or even any) relevant documents. It is surprising that the greatest improvement can be seen when 10 collections are selected (instead of 15 or 20). This can be seen in both Table 3.8 and Figure 3.6. This may be explained by a phenomenon – there are queries for which many relevant documents can be found in the top 10 collections. For these queries, searching a larger number of collections does not provide a large benefit.

Searching additional collections tends to improve retrieval performance, but there are limits to that trend. In fact, beyond a certain point, searching additional collections may degrade performance.

REUTER-TEST 200-collection testbed				
Precision at *s* docs (% change)	**Collections 20**	**Collections 15**	**Collections 10**	**Collections 5**
10	0.4437 (-6.03)	0.4611 (-2.35)	0.4778 (+1.18)	0.4722
20	0.4500 (-1.81)	0.4611 (+0.61)	0.4694 (+2.42)	0.4583
30	0.4354 (-3.75)	0.4537 (+0.28)	0.4648 (+2.72)	0.4524
40	0.4359 (-2.83)	0.4527 (+0.91)	0.4569 (+1.85)	0.4486
50	0.4375 (+2.29)	0.4488 (+4.93)	0.4544 (+6.24)	0.4277
60	0.4302 (+7.30)	0.4407 (+9.92)	0.4546 (+13.39)	0.4009
70	0.4205 (+10.57)	0.4358 (+14.59)	0.4499 (+18.3)	0.3803
80	0.4179 (+9.88)	0.4384 (+15.27)	0.4432 (+16.53)	0.3803
90	0.4167 (+20.57)	0.4273 (+23.64)	0.4370 (+26.44)	0.3456
100	0.4150 (+28.16)	0.4112 (+26.99)	0.4250 (+31.25)	0.3238

Table3.8: The effect of selection collection size on retrieval performance in the REUTER-TEST

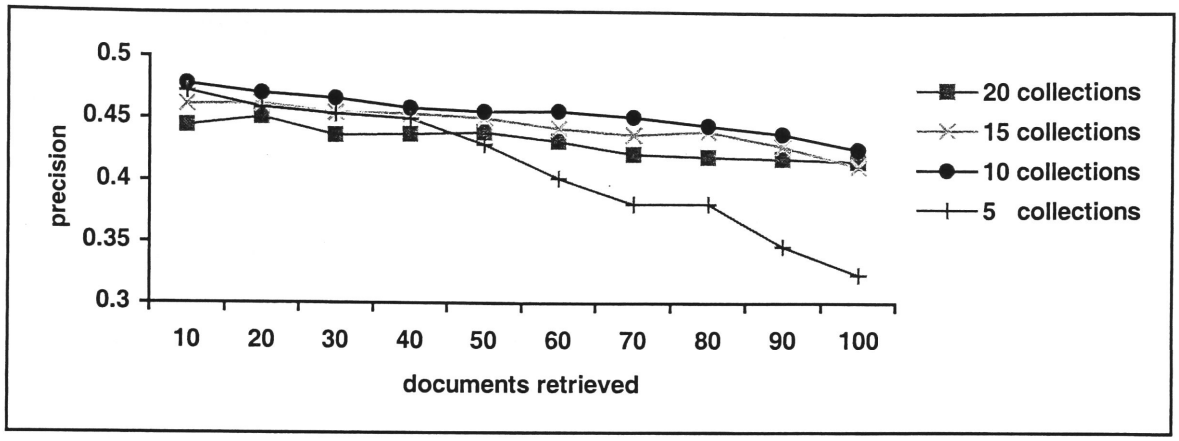

Figure 3.6: The effect of selection collection size on retrieval performance in the REUTER-TEST

3.4.3.3 Training collections

Large training sets are required to provide a useful classification and to get accurate expansion concepts for *Naive Bayes* learning. Since it is tedious and expensive to create these sets of labeled data, we naturally consider the impact of using smaller training collections. So instead of using the full REUTER-TRAIN collections for query expansion, we vary the amount of labeled training data by 75%, 50% and 25% of the REUTER-TRAIN to get expansion concepts.

Table 3.9 and Figure 3.7 show retrieval results. It understands that the full TRAIN has the best performance at a large labeled data. There is a rapid decrease in performance as 25% of the labeled data in TRAIN is used. Comparing with using full TRAIN, there is only a small degradation, especially at higher cut-offs (about 2.4%) when 75% and 50% of TRAIN are used for query expansion. It suggests that it is possible to cut the size of the training collection without significantly affecting retrieval effectiveness. However, currently we do not know how to automatically

determine the optimal size of the training set. I need to do the further investigations to solve the problem in my future research.

REUTER-TEST 200-collection testbed				
Precision at *s* docs (% change)	**TRAIN**	**TRAIN 75%**	**TRAIN 50%**	**TRAIN 25%**
10	0.465	0.4450 (-4.3)	0.4450 (-4.3)	0.3600 (-22.58)
20	0.4525	0.4250 (-6.07)	0.4200 (-7.18)	0.3575 (-20.99)
30	0.4416	0.4083 (-7.54)	0.4050 (-8.28)	0.3333 (-24.52)
40	0.4275	0.3975 (-7.01)	0.3962 (-7.32)	0.3225 (-24.56)
50	0.4220	0.3970 (-5.92)	0.3950 (-6.39)	0.3180 (-24.64)
60	0.4174	0.3942 (-5.55)	0.3941 (-5.58)	0.3124 (-25.15)
70	0.4042	0.3885 (-3.88)	0.3941 (-2.94)	0.3085 (-23.67)
80	0.3943	0.3781 (-4.10)	0.3849 (-2.38)	0.3031 (-23.12)
90	0.3821	0.3644 (-4.61)	0.3731 (-2.35)	0.3011 (-21.19)
100	0.3720	0.3535 (-4.97)	0.3731 (+0.29)	0.3005 (-19.22)

Table3.9: The effect of the training collection size on retrieval performance in the REUTER-TEST

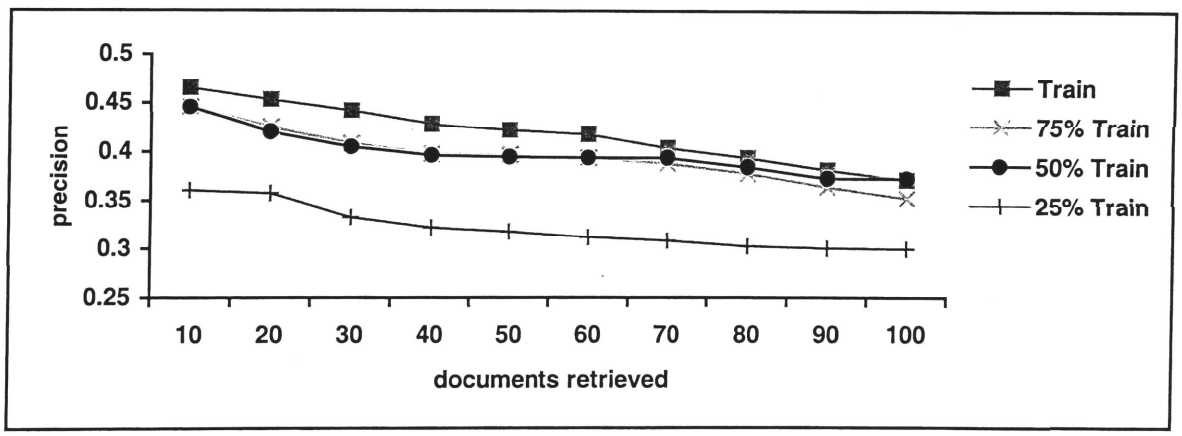

Figure 3.7: The effect of the training collection size on retrieval performance in the REUTER-TEST

3.4.3.4 Weight of expansion concepts

The high baseline of the REUTER-TEST data set (46.3% average precision) suggests that the original queries are of very good quality and we should give them more

emphasis. So, we add a parameter that varies the relative contribution of expansion concepts on retrieval performance. Table 3.10 and Figure 3.8 show that downweighting the expansion concepts does improve performance. Experiments are conducted with weight values ranging from 0.2 to 1. The results indicate that when we downweight the expansion concepts by 80% by reducing the weight of query from 1 to 0.2, the retrieval performance is slightly better than other weight values. It suggests that although expansion concepts help to improve retrieval effectiveness, we should pay more attention on the base query in case that improper expansion concepts hurt retrieval performance.

REUTER-TEST 200-collection testbed					
Precision at *s* docs (% change)	**Weight 1**	**Weight 0.8**	**Weight 0.6**	**Weight 0.4**	**Weight 0.2**
10	0.4800	0.4800 (0)	0.4850 (+1.04)	0.4850 (+1.04)	0.4900 (+2.08)
20	0.4825	0.4805 (-0.41)	0.4850 (+0.51)	0.4900 (+1.55)	0.4925 (+2.07)
30	0.4800	0.4817 (+0.35)	0.4800 (0)	0.4850 (+1.04)	0.4883 (+1.73)
40	0.4712	0.4737 (+0.53)	0.4777 (+1.38)	0.4825 (+2.39)	0.4887 (+3.71)
50	0.4690	0.4710 (+0.42)	0.4750 (+1.27)	0.4830 (+2.98)	0.486 (+3.62)
60	0.4668	0.4683 (+0.32)	0.4701 (+0.7)	0.4750 (+1.75)	0.4817 (+3.19)
70	0.4694	0.4657 (-0.78)	0.4671 (-0.4)	0.4750 (+1.19)	0.4786 (+1.95)
80	0.4587	0.4726 (+3.03)	0.4650 (+1.37)	0.4699 (+2.44)	0.4750 (+3.55)
90	0.4511	0.4533 (+0.48)	0.4556 (+0.99)	0.4662 (+3.34)	0.4694 (+4.05)
100	0.4395	0.4430 (+0.79)	0.4475 (+0.11)	0.4662 (+6.07)	0.4625 (+5.23)

Table3.10: The effect of the different weight of expansion concepts on retrieval performance in the REUTER-TEST

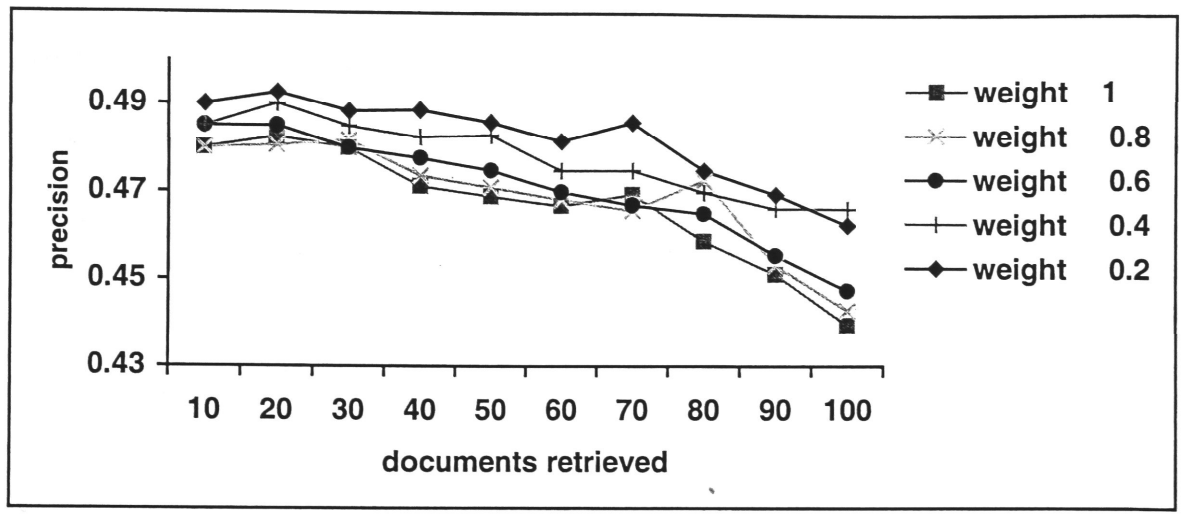

Figure 3.8: The effect of the different weight of expansion concepts on retrieval performance in the REUTER-TEST

3.5 Summary

In this chapter, we emphasized on introducing the work principle of the three major components in ***IISS*** system. Analysis Agent is developed by a *Naive Bayes classifier* to expand the user's query with other related terms or concepts in order to make the expanded query more suitable for the information source selection. Case Matching Agent uses *Case-based Reasoning* algorithm to select a set of promising information sources to search based on the confidence factors for information source with respect to every term in the query. An adaptation algorithm – *reinforcement learning* is used by Learning Agent to learn the user's feedback in order to improve the quality of search results and adapt to the frequent changes of the dynamic environment.

We have finished the implementation of Analysis Agent and investigated the effect of using query expansion for both collection selection and retrieval on the **Reuters 21578** data set.

Analysis of the experiment results supports the following conclusions:

- When query expansion with a *Naive Bayes classifier* is employed, selection performance will be better than that of using the base query.
- Query expansion does improve retrieval performance if the number of expansion concepts is chosen properly.
- It is possible to achieve good retrieval performance by selecting more collections (up to a point).
- Large training sets are required to provide useful classification and to get accurate expansion concepts for *Naive Bayes* learning. The results suggest that it is possible to cut the size of the training collection without significantly affecting retrieval effectiveness.
- To avoid improper expansion concepts to hurt retrieval performance, it is necessary to downweight the expansion concepts.

Chapter 4

Conclusions and future work

4.1 Conclusions

The use of information retrieval systems in distributed information environments raises a new set of issues that have received widely attention. These issues include selecting the best set of collections from a ranked list, processing the query on the selected collections and produce a set of individual result-lists, and merging the document rankings that are returned from a set of collections. The interesting problems of Query Process and Result Merging are beyond the scope of the present work, where the focus is on Information Source Selection.

As we described in Chapter 1, the research in this thesis is directed at investigating how to select the most promising information sources under a distributed information environment to search using intelligent agent technique.

IISS is an agent-based intelligent information source selection system for distributed information sources. This is an ongoing project, and we plan to enhance ***IISS*** by expanding its capabilities.. To this point, our contributions are to propose an intelligent environment where the AA, CMA and LA, these three major agents iteratively work together to locate the most appropriate information sources to search so as to effectively and efficiently satisfy the user' expectations. Each agent performs a special task so as to ease the complexity of information source selection. Three

artificial intelligent techniques, *Naive* Bayes Classifier, case-based reasoning and reinforcement learning in this integrated system, are described in detail. A *Naive Bayes* text classifier is used to facilitate query expansion, the CBR-IR approach is designed to locate relevant information sources and reinforcement learning algorithm is developed to be adaptive to changing information source performance.

Currently, we have finished the implementation of the first component – Analysis Agent in ***IISS*** system. The experimental results on the **Reuters 21578** data set are extremely encouraging. They suggest that it is possible to improve the effectiveness on both selection and retrieval stages in distributed searching environments by using query expansion with a *Naive Bayes classifier.*

4.2 Future work

However, there are a number of areas in which we will continue our work. Firstly, it is important for us to continue to implement other two major agents – Case Matching Agent and Learning Agent so that we can get the final evaluation results of the whole system. Secondly, we plan to use even larger collections such as the 20 Gigabytes TREC VLC (Very Large Corpus) collection to test our techniques. Thirdly, we try to find a "versatile" training collection for query expansion. Such a collection should have a wide coverage of subject matters so that most queries can be properly expanded.

BIBLIOGRAPRAPHY

[Bau 97] C. Baumgarten. A Probabilistic Model for Distributed Information Retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Philadelphia, PA, 1997, pp. 258-266,

[CLC 95] J. P. Callan, Z. Lu, and W. B. Croft. Searching Distributed Collections with Inference Networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Seattle, 1995, pp. 21-29.

[CCH 92] J. P. Callan, W. B. Croft and S. M. Harding. The INQUIRY Retrieval System. In P*roceedings of the Third International Conference on Database and Expert System Application,* Valencia, Spain, 1992, pp. 78-83.

[CM 1996] B. Cahoon and K. S. McKinley. Performance Evaluation of a Distributed Architecture for Information Retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, 1996, pp. 110 – 118.

[CP 2000] J. Callan, A, L, Powell. The Effects of Query-based Sampling. *Technique Report LMU-LTI-00162, Lauguage Technologies Institute,* School of Computer Science, Carnegie Mellon University. 2000.

[DAN 91] P. B. Danzig, J. Ahn, J. Noll and K. Obraczka. Distributed Indexing: A Scalable Mechanism for Distributed Information Retrieval. In *Proceedings of the 14th*

Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Chicago, Illinois, 1991, pp. 220 – 229.

[DK 97] B. Das and D. Kocur. Experiments in Using Agent-based Retrieval from Distributed and Heterogeneous Databases. In *Proceedings of Knowledge and Data Engineering Exchange Workshop,* NewPort Beach, CA, 1997, pp. 27 –35.

[DLP 95] K. Decker, V. Lesser, M. V. Nagendra Prasad & T. Wagner. MACRON: An Architecture for Multi-agent Cooperative Information Gathering. In *Proceedings of the CIKM Workshop on Intelligent Information Agents.* Baltimore, MD, 1995, pp. 1-11.

[EW 94] O. Etzioni, and D. S. Weld. A Softbot-Based Interface to the Internet. In *Communications of the ACM*, **37(7)**, July 1994, pp. 72-76.

[Fuhr 99] N. Fuhr. A Decision-Theoretic Approach to Database Selection in Networked IR. In *ACM Transactions on Information Systems,* **17(3),** 1999, pp. 229-249.

[FPC 99] J. C. French, A. L. PoweU, J. Callan, C. L. Viles, T. Emmitt, K. J. Prey, and Y. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Berkekey,1999, pp. 238-245.

[FPV 98] J. C. French, A. L. Powell, C. L. Viles, T. Emmitt, and K. J. Prey. Evaluating Database Selection Techniques: A Testbed and Experiment. In *Proceedings of the 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Melbourne, Australia, 1998, pp. 121-129.

[GGT 94] L. Gravano, H. Garcia-Molina, and A. Tomasic. The Effectiveness of GLOSS for the Text Database Discovery Problem. In *Proceedings of International*

Conference on Management of Data and Symposium on Principle of Database System, Philadelphia, 1994, pp. 126 – 137.

[HT 99] D. Hawking and P. Thistlewaite. Methods for Information Server Selection. In *ACM Transaction on Information System* **17(1)**, 1999, pp. 40 – 76.

[KL 96] L. P. Kaelbling, M. L. Littman and A. W. Moore. Reinforcement Learning: A Survey. In *Journal of Artificial Intelligence Research,* 1996, pp. 237-285.

[Lew 97] D. D. Lewis. Reuter-21578 Text Categorization Test Collection Distribution 10. Available at http://www.research.att.com/~lewis.

[Lew 98] D. D. Lewis. *Naive* Bayes at Forty: The Independence Assumption in Information Retrieval. In *Proceedings of the 10th* European Conference Machine Learning, Chemnitz, Germany, 1998.

[MN 98] Andrew McCallum and Kamal Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization,* 1998.

[Mul 94] M. E. Muller. An Intelligent Multi-Agent Architecture for Information Retrieval from the Internet. In http: // citeseer.nj.nec.com/ cachedpage/94069/1

[MZ 95] A. Moffat and J. Zobel. Information Retrieval Systems for Large Document Collections. In *Proceedings of the Third Text Retrieval Conference,* Maryland, 1995.

[Por 80] M. Porter. An Algorithm for Suffix Stripping . In *Program, **14***, 1980.

[RS 89] C. K. Risesbeck and R. C. Schank. Inside Cased-based Reasoning. *Laurence Erlbaum Associates Publishers,* New Jersey, 1989.

[SM 83] G. Salton and M. Mcgill. Introduction of Modern Information Retrieval. *McGrag-Hill Publisher, 1983.*

[SEK 92] M. Schwartz, A. Emtage, B. Kahle, and B. Neumann. A Comparison of Internet Resource Discovery Approachs. In *Computer Systems,* **5(4),** 1992, pp. 461-493.

[SPW 96] K. Sycara, A. Pannu, M. Willamson, D. Zeng; K. Decker. Distributed Information Agents. In *IEEE Expert,* **11(6),** Dec. 1996, pp. 36 –46.

[VGJ 95] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. The Collection Fusion Problem. In *Proceedings of the Third Text Retrieval Conference,* Maryland, 1995, pp. 95-104.

[XC 99] J. Xu and W. B. Croft. Cluster-based Language Models for Distributed Retrieval. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Berkeley,1999, pp. 254-261.

[YR 98] R. R. Yager and A. Rybalov. On the Fusion of Documents from Multiple Collection Information Retrieval Systems. In *Journal of the American Society of Information Science,* **49(13),** 1998, pp. 1177-1184.

[YHM 98] J. Yang, V. Honavar, L. Miller and J. Wong. Intelligent Mobile Agent For Information Retrieval and Knowledge Discovery from Distributed Data and Knowledge Sources, In http://www.cs.iastate.edu/~ honavar/papers/it98-moblies.ps.

[YL 97] B. Yuwono and D. L. Lee. Server Ranking for Distributed Text Retrieval Systems on Internet. In *Proceeding of* the *Conference on Database Systems for Advanced Applications,* New Orleans, 1997, pp. 41- 49.

[Zhang 98] M. Zhang. A Case-based Strategy for Solution Synthesis among Cooperative Expert Systems. In *Systems Formalisms, Methodologies and*

Applications, Lecture Notes in Artificial Intelligence, **1441**, Springer Verlag Publishers, 1998, pp. 231-240.